software modularity group

a fundamental breakthrough in how we think about and work in software development -- *the 'multiple view thing' is about to take off in a big way*

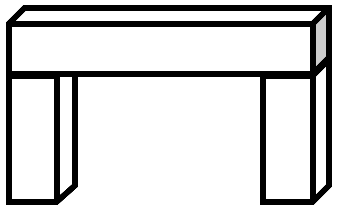
Gregor Kiczales
University of British Columbia
PARC

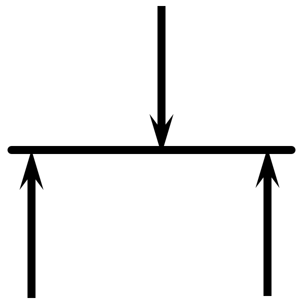
a fundamental breakthrough is possible

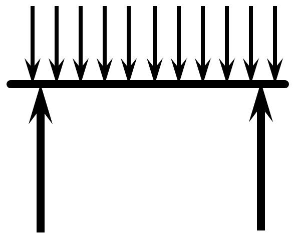
- **a diversity of models is
  a key enabler for all engineering fields**
  - hierarchical models
  - <u>and</u> crosscutting models

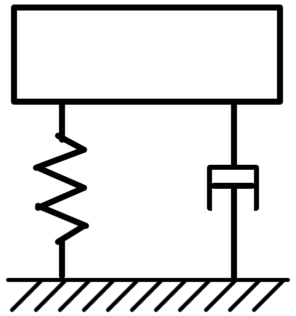
- **new result crosscutting programs**
  - today
    - can have significant impact on industrial SDP
  - tomorrow
    - can marry the best of
      - language/programming based approaches (scruffies)
      - formal/model-based approaches (neats)
    - key support for “science of engineering of software”

models – hierarchical and crosscutting


artifact

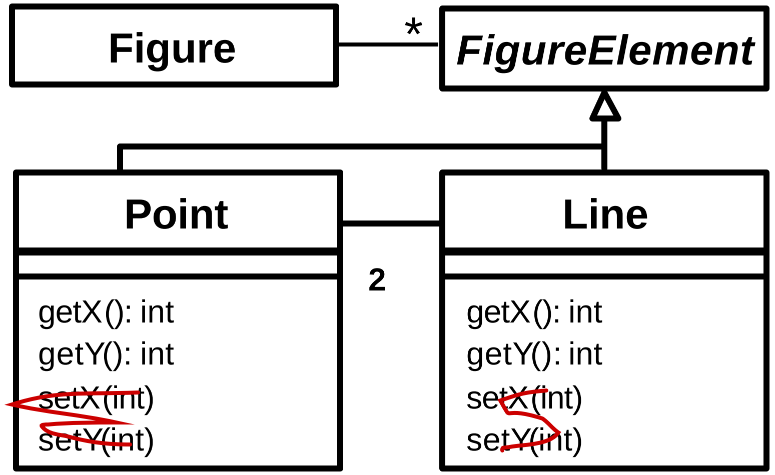

simple statics


more detailed statics


simple dynamics

- **dynamic model crosscuts static models**
- **intuitively crosscutting means:**
  something spread-out in one view
  is local in the other, and vice versa

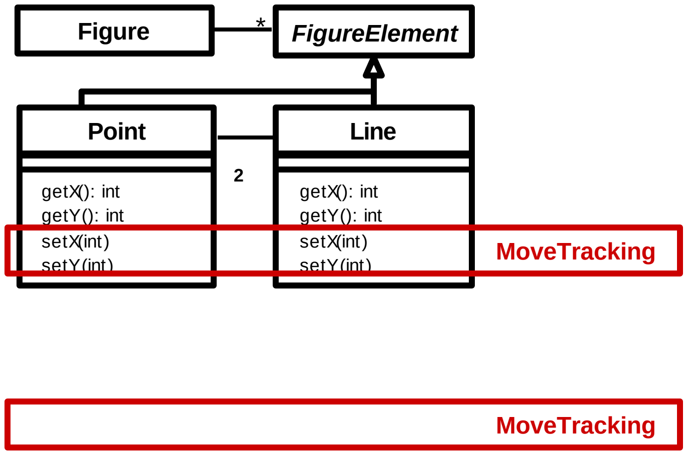
programs – hierarchical or level of detail



```
class Line {
  private Point p1, p2;

  Point getP1() { return p1; }
  Point getP2() { return p2; }

  void setP1(Point p1) {
    this.p1 = p1;
  }
  void setP2(Point p2) {
    this.p2 = p2;
  }
}

class Point {
  private int x = 0, y = 0;

  int getX() { return x; }
  int getY() { return y; }

  void setX(int x) {
    this.x = x;
  }
  void setY(int y) {
    this.y = y;
  }
}
```

move tracking

crosscutting programs



```
class Line {
  private Point p1, p2;

  Point getP1() { return p1; }
  Point getP2() { return p2; }

  void setP1(Point p1) {
    this.p1 = p1;
  }
  void setP2(Point p2) {
    this.p2 = p2;
  }
}
```

```
class Point {
  private int x = 0, y = 0;

  int getX() { return x; }
  int getY() { return y; }

  void setX(int x) {
    this.x = x;
  }
  void setY(int y) {
    this.y = y;
  }
}
```

```
aspect DisplayUpdating {

  pointcut move():
    call(void Line.setP1(Point))          ||
    call(void Line.setP2(Point))          ||
    call(void Point.setX(int))            ||
    call(void Point.setY(int));


  after() returning: move() {
    Display.update();
  }
}
```

an aspect is a modular unit of crosscutting code (and design)

impact and agenda

- **0-3 years**
  - early adopter Java programmers
    - improved productivity, configurability, adaptability…
    - using aspect-oriented programming
      - AspectJ, Hyper/J, Demeter…
    - using aspect-oriented software development
      - connections to UML, IDEs
  - research work
    - security, robustness, distribution…
    - language design and implementation
    - tools support, methods, processes, refactoring
  - beginning to see transition to industry
    - but must maintain research

impact and agenda

- **3-10 years**
  - ability to have crosscutting programs enables
    - design rationale capture to align with code
    - models to align with code
      - ‘rountrip’ between model and code level
      - eliminate win-lose formal/programming struggle
    - technological basis for ‘engineering of software’
      - we can support multiple
  - wonderful work to do
    - are now seeing burst of proposals for different views
    - develop a science of this
      - what will our time–frequency domain transform be?
    - build a new community structure